ITI0101 Sissejuhatus infotehnoloogiasse.

04. Web Applications.

Martin Verrev
martin.verrev@taltech.ee

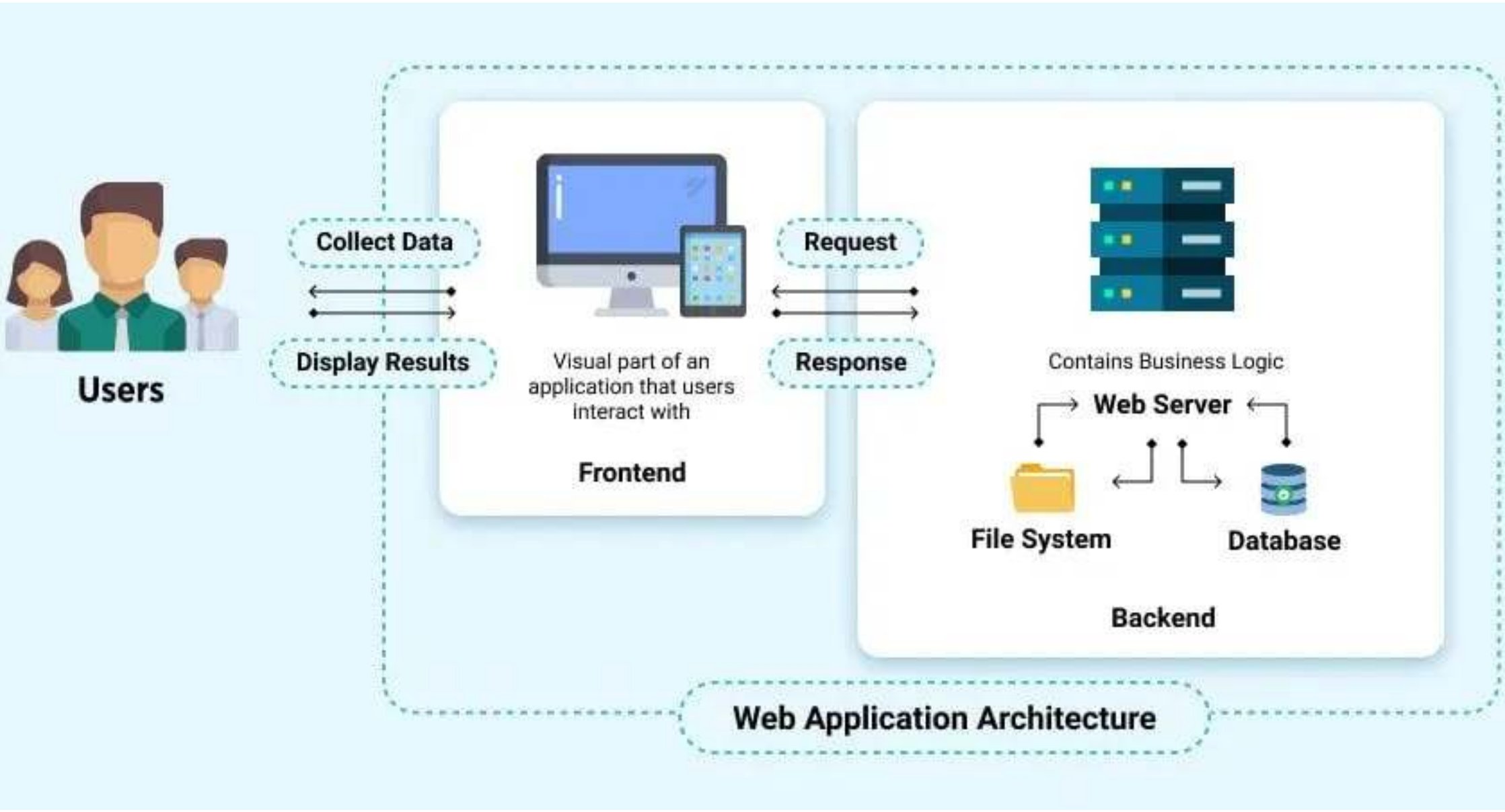
Web applications are software programs that are accessed over the internet through a browser. They are designed to be run on a web server and can be accessed by any device that has an internet connection, including smartphones, tablets, and laptops.

Benefits

1. Cross-Platform Compatibility
2. Ease of of access
3. Reduced costs for users
4. Streamlined updates and maintenance
5. Scalability
6. Centralized data management
7. Integration with other applications and data

Disadavantages

1. Dependence on internet connectivity
2. Performance limitations
3. Browser compatibility issues
4. Security vulnerabilities
5. Limited access to device features


Users
Collect Data
Display Results
Visual part of an application that users interact with
Frontend
Request
Response
Contains Business Logic
Web Server
File System
Database
Backend
Web Application Architecture

Web application consists of:

1. **Front-end *(esirakendus)*:** This is the part of the web application that the user interacts with, such as the user interface (UI) and user experience (UX) design. It usually consists of HTML, CSS, and JavaScript, which are used to create the visual elements and enable user interaction.
2. **Back-end: *(tagarakendus)*** This is the part of the web application that runs on the server and is responsible for processing data and generating responses to client requests. It includes server-side scripting languages such as PHP, Python, or Ruby, and databases such as MySQL or PostgreSQL.

Web application consists of:

3. **Web server:** The web server is responsible for receiving and responding to HTTP requests from clients. Popular web servers include Apache, Nginx, and IIS.
4. **APIs:** Application Programming Interfaces (APIs) allow different software components to communicate with each other. Web applications may use APIs to integrate with other services or to enable third-party developers to build applications that interact with their data.
5. **Security:** Security is a critical component of any web application, as it helps protect user data and prevent unauthorized access. This includes measures such as SSL/TLS encryption, secure authentication, and access control

How does it work (1/3)

- (1) The user enters the URL for the web application or clicks a link to access it in a web browser.
- (2) The web browser sends a request to the web server. The request contains information such as the HTTP method (e.g., GET, POST), the requested URL, and any additional data.
- (3) The web server receives the request and determines the appropriate handler or controller based on the requested URL. This routing process is often managed by a web framework or server-side scripting language.
- (4) Once the web server identifies the appropriate handler, it processes the request. This may involve fetching data from a database, performing computations, or executing various tasks based on the nature of the request.

How does it work (2/3)

- (5) The web application’s business logic handles the processing of the request. It includes tasks such as handling user input, performing calculations, and interacting with databases or external systems.
- (6) If the web application needs to retrieve or store data, it interacts with a database. User information, application data, and other relevant data are stored and managed in the database. A database management system (DBMS) like MySQL, PostgreSQL, or MongoDB may be used for data operations.
- (7) Once the server has processed the request and obtained the necessary data, it generates a response. The response typically includes HTML, CSS, JavaScript, and other resources required to render the web page.
- (8) The web server sends the response back to the user’s web browser.

How does it work (3/3)

- (9) The user’s web browser receives the response and renders the web page based on the received HTML, CSS, and JavaScript.
- (10) The user can now interact with the web application by clicking buttons, filling out forms, or performing other actions.
- (11) User interactions trigger additional requests to the server, initiating a new cycle of processing. These requests follow the same client-server architecture and request-response cycle.

Two ways for a server to get input from a user in a web application:

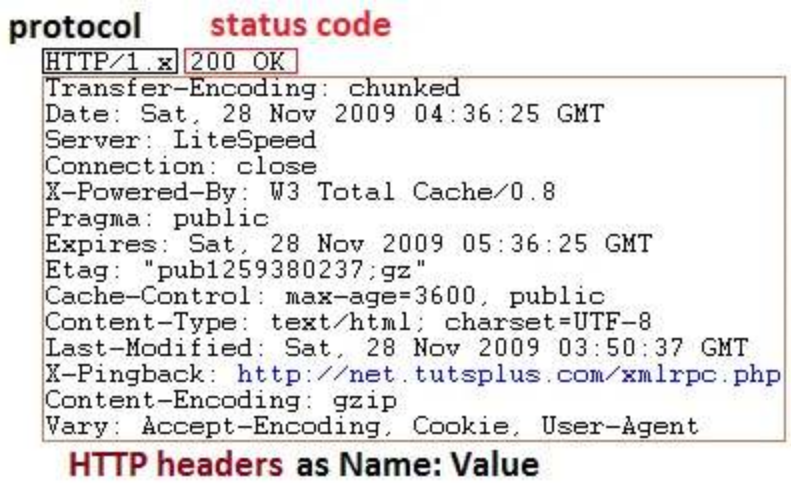
- **Header/URL Parameters:** In this method, input is passed to the server through the URL of the requested web page. This is typically done by appending parameters to the end of the URL, separated by question marks and ampersands.
- **Body:** The HTTP body, also known as the message body, is part of an HTTP request or response that carries the actual data being sent or received. It is located after the headers in an HTTP message and is separated from the headers by a blank line.

HTTP Request

HTTP request headers are sent by the client (typically a web browser or an application) to the server to provide additional information about the request.

HTTP Response

HTTP response headers are sent by the server back to the client as part of the HTTP response. They provide information about the response and additional instructions to the client.



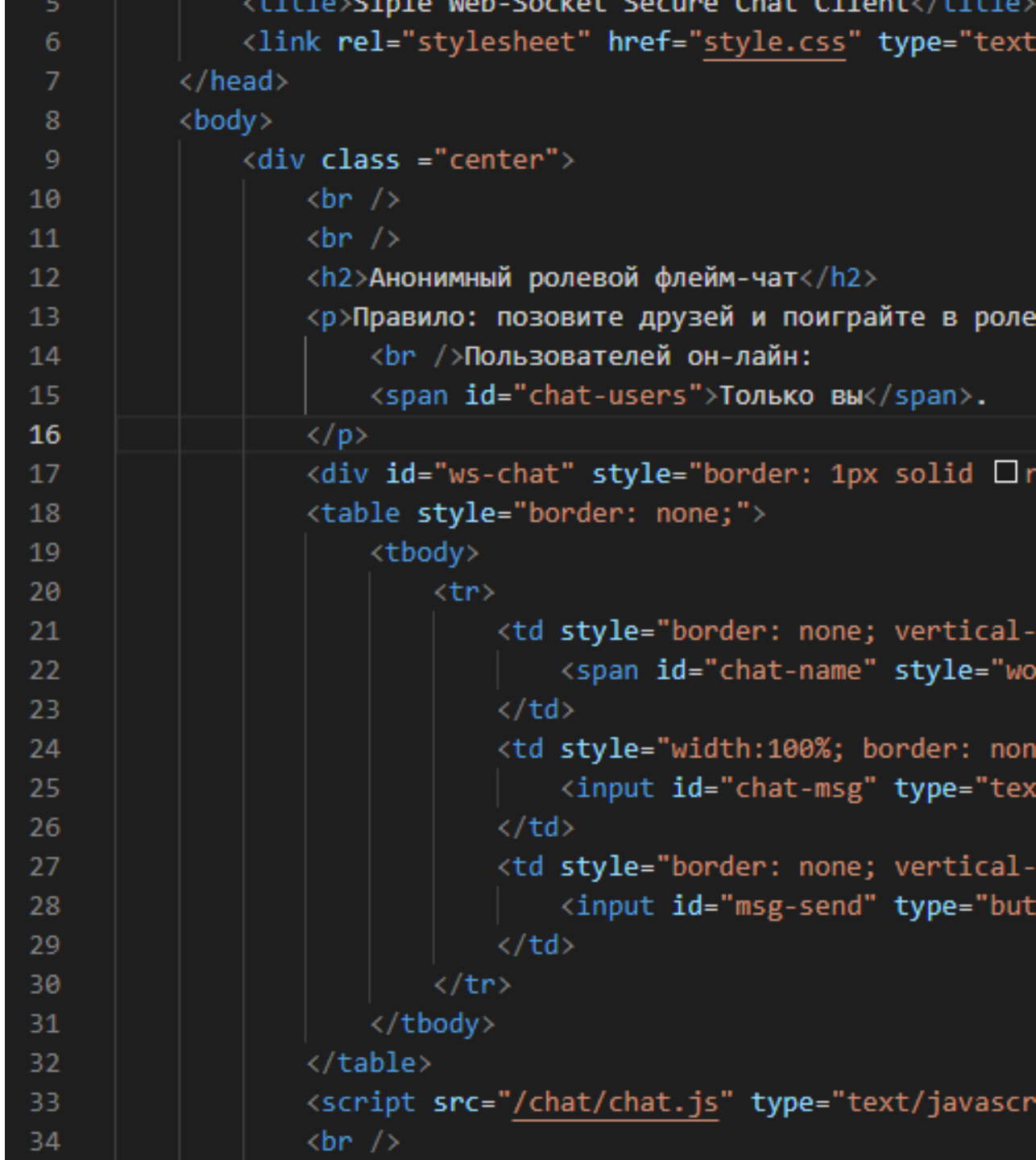
Technologies: Frontend

HTML

Most of the stuff on the web is no different than the stuff on your computer - it is just a whole load of files sorted into a whole load of directories.

HTML files are nothing more than simple text files, so to start writing in HTML, you need nothing more than a simple text editor.

https://htmldog.com/guides/html/beginner/gettingstarted



Fragment from first HTML page in existence (1993)

```
<HEADER>
<TITLE>The World Wide Web project</TITLE>
<NEXTID N="55">
</HEADER>
<BODY>
<H1>World Wide Web</H1>The WorldWideWeb (W3) is a wide-area<A
NAME=0 HREF="WhatIs.html">
hypermedia</A> information retrieval
initiative aiming to give universal
access to a large universe of documents.
```

Source: https://info.cern.ch/hypertext/WWW/TheProject.html

See also: https://first-website.web.cern.ch/first-website/

Different versions until 2008 and then:

https://www.hostinger.com/tutorials/wp-content/uploads/sites/2/2017/03/milestones-of-HTML.jpg

See also: Browser Wars

```
<!DOCTYPE html>
<html>
    <head>
        <title>HTML Page Sample</title>
        <meta charset="UTF-8">
    </head>
    <body>
        Lehe sisu
    </body>
</html>
```

HTML: Elements, tags, attributes

In general tags are closed: `<h1>Tere Maailm</h1>`

.. but as always there are exceptions: `<br>`, `<meta>`, `<img>`, `<input>` ...

HTML5 Tags/Elements

HTML tags by purpose

- Describing structure: `<a>`, `<body>`, `<h1>` kuni `<h6>` ...
- Describing metadata: `<title>`, `<meta>`, `<base>` ...
- Describing input: `<form>`, `<input>`, `<select>` ..
- Formatting content: `<code>`, `<sup>`, `<strong>` ..
- Describing lists: `<ul>`, `<li>` ..
- Describing tables: `<table>`, `<thead>`, `<td>` ..
- Controlling scripts: `<script>` ja `<noscript>`
- Presenting content: `<img>`, `<canvas>`, `<iframe>`

Let's try

Recommendations for writing HTML

- Always set document type to HTML5 **`<!DOCTYPE html>`**
- Always set text encoding to UTF-8 **`<meta charset="UTF-8">`**
- Always use **lowercase** to define elements: **`<p>`** not `<P>`
- Always quote the attributes `<img src="images/1.png" alt="Cat photo">`
- Use double quotes: **`"`** not **`'`**.
- Write full file extension: **`index.html`** not `index.htm`
- Always use **lowercase** file extensions: **`index.html`** not `Index.html`, **`user-data.html`** mitte `userData.html`

Cascading Style Sheets **(CSS) exists to style your your content. It is somewhat different to HTML but it remains simple and straightforward.**

In fact, it has been a constant source of delight for me over the past year to get to continually tell hordes (literally) of people who want to – strap yourselves in, here it comes – control what their documents look like in ways that would be trivial in TeX, Microsoft Word, and every other common text processing environment:
"Sorry, you're screwed."

— Marc Andreessen, 1994

```
html {
  margin-left: 2cm;
  font-family: "Times", serif;
}

h1 {
  font-size: 24px;
}
```

A fragment of CSS from 1996 written by Håkon Wium Lie - author of standardized CSS.

See also: A Look Back at the History of CSS.
https://css-tricks.com/look-back-history-css/

Using CSS

- Embedded to element: *(inline)*
- Embedded to document: *(internal)*
- Separate style file

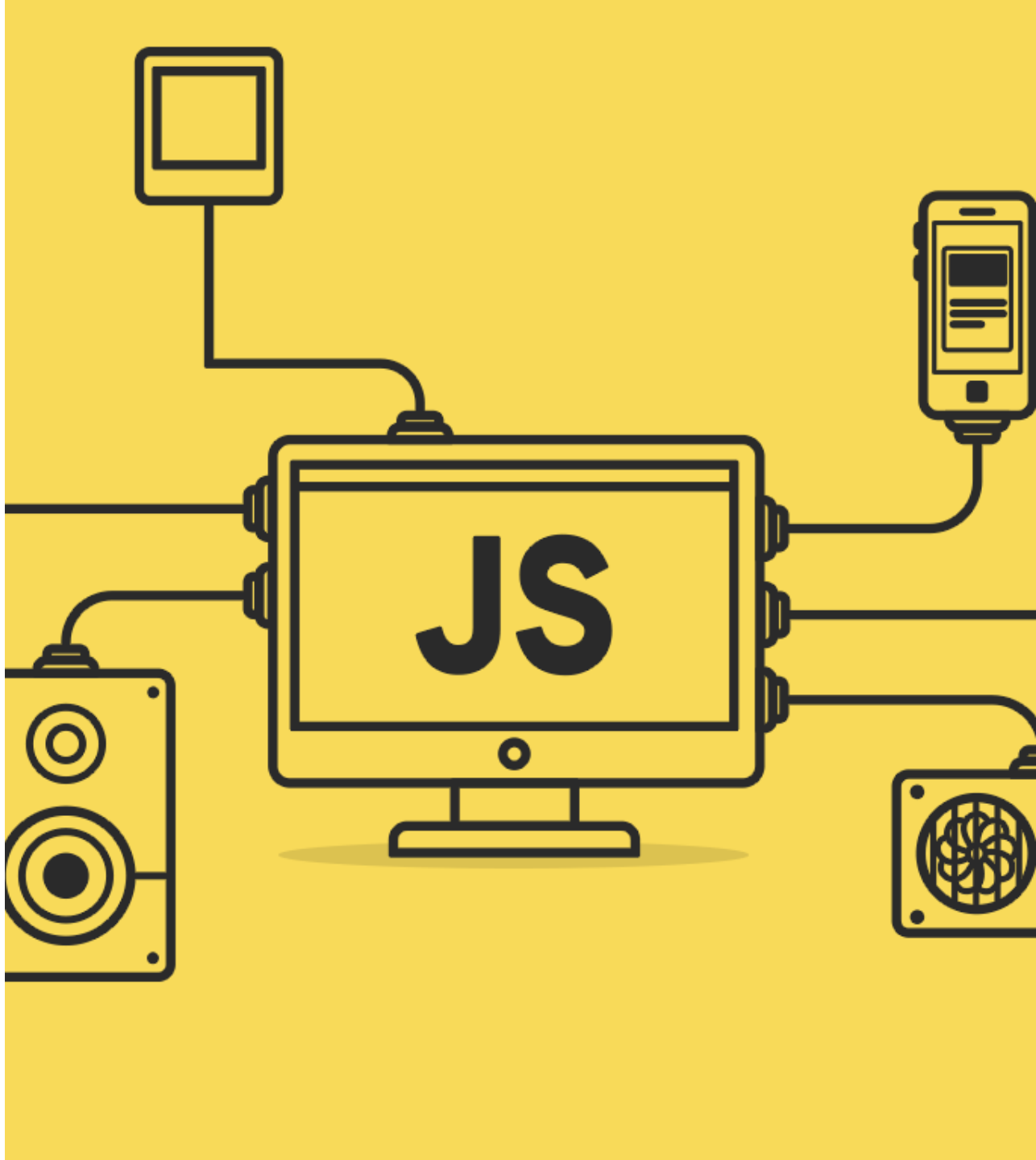
CSS Syntax

JavaScript is the only language that I'm aware of that people feel they don't need to learn before they start using it.

Douglas Crockford

Javascript

- JavaScript was invented by Brendan Eich in 1995, and became an ECMA standard in 1997.
- ECMAScript is the official name of the language.
- Latest version: ES6. See also: https://medium.com/@diwakarkashyap/es5-vs-es6-in-javascript-14-big-changes--3e19caab625a



What can be done with Javascript?

- Change page content and styles
- Respond to user mouse clicks, key presses and other actions.
- Send requests to other servers, upload and download files.
- Show notifications to the user and ask the user for various information.
- Remember user data (*local storage*)
- Interact with the user's webcam and microphone

..

And much more

What cannot be done with Javascript?

- Read and write files from the hard drive
- Read information from other browser windows, tabs and applications

How to use

- At the element
- Between `<script>` tags in the header or body of the document.
- Linked as an external file

```
<!DOCTYPE html>
<html>
    <head>
        <title>HTML Page Sample</title>
        <meta charset="UTF-8">
    </head>
    <body>
        <h1>Lehe sisu</h1>
        <script src="js/skript.js"></script>
    </body>
</html>
```

Variables

- Variable must be initialized before use
- `let` keyword for initialization, e.g. `let name="Bob";`
- `const` to initialize constants, eg `const MAX_TURNS=3;`
- Also `var` from earlier versions, eg `var name="John";`. It is not directly wrong, but allows to rewrite already written variables.

HTML DOM

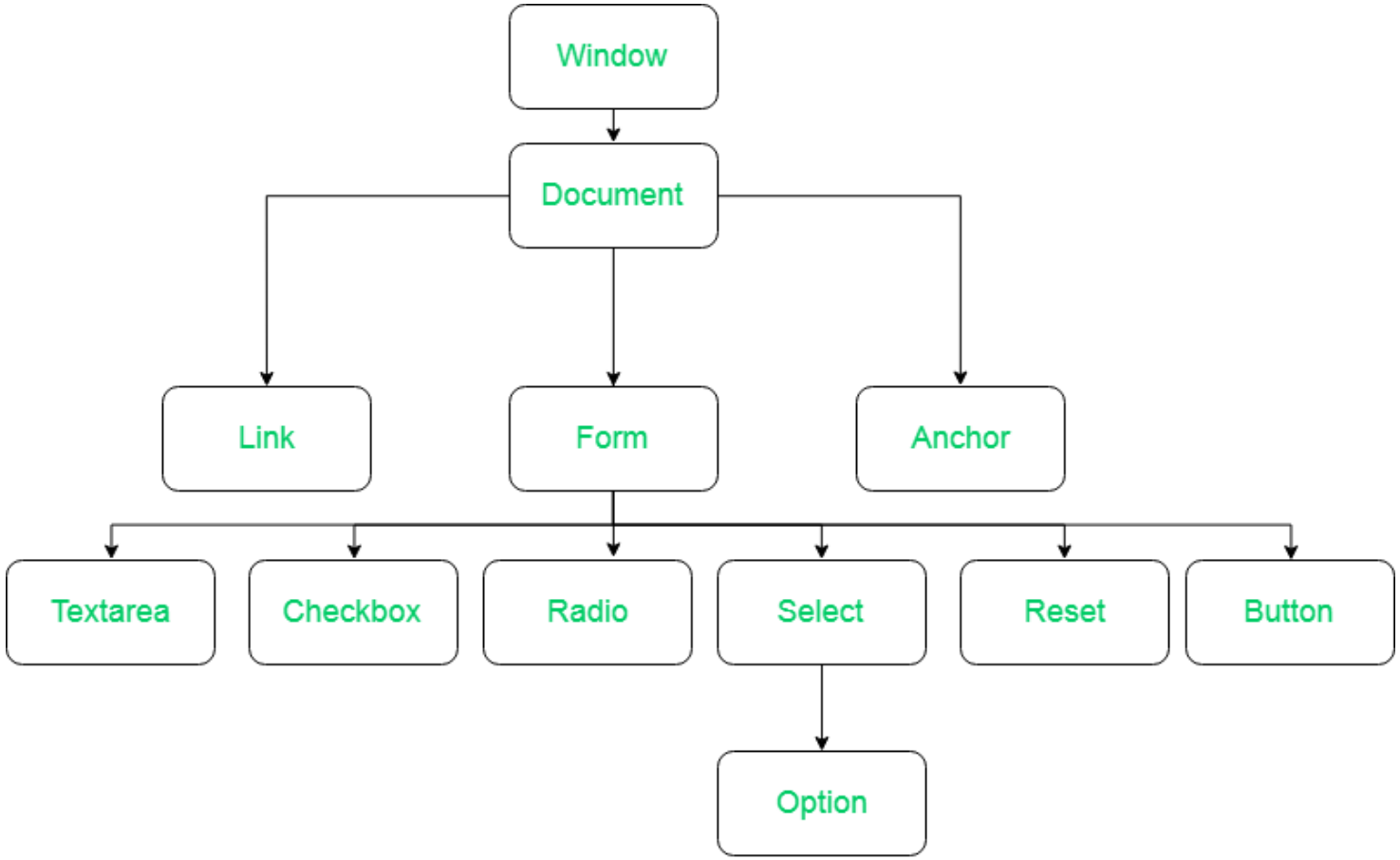
The **HTML DOM** is an **Object Model** for **HTML**. It defines:

- HTML elements as **objects**
- **Properties** for all HTML elements
- **Methods** for all HTML elements
- **Events** for all HTML elements

The **HTML DOM** is an **API** (Programming Interface) for **JavaScript**:

- JavaScript can add/change/remove HTML elements
- JavaScript can add/change/remove HTML attributes
- JavaScript can add/change/remove CSS styles
- JavaScript can react to HTML events
- JavaScript can add/change/remove HTML events

Document Object Model (DOM)



Vaata ka: https://www.w3schools.com/JSREF/dom_obj_document.asp

Accessing Elements via DOM

By ID, class name or type

- `document.getElementById(elem)`
- `document.getElementsByClassName(elem)`
- `document.getElementsByTagName(tag)`

Universal Search

- `document.querySelector(search)`
- `document.querySelectorAll(search)`

Manipulating elements via DOM

- `let elem = document.createElement(type)`
- `parent.appendChild(elem)`
- `parent.removeChild(elem)`

Events

- HTML `<element event="some JavaScript">`
- Javascript `element.onclick = somefunction()`
- Event handler `addEventListener("click", someFunction);`

HTML APIs are a collection of JS libraries which can be used directly in HTML files without incorporating any customize JavaScript code

https://medium.com/the-ui-girl/html-apis-in-depth-78f0abc918c8

Example Uses

- **Integration with other services**: payments, credentials, offline mode ...
- **Input**: Touch, speech recognition ...
- **Camera & Microphone**: audio and video capture, recording, shape detection ...
- **Native Behaviors**: push notifications, permissions, task scheduling ...
- **Surroundings**: NFC, bluetooth, ambient light ...
- **Device Status**: Network status, battery status ...
- **Location and Position**: Geolocation, device motion ...
- **Screen and Output**: Fullscreen, VR, screen orientation ...

https://whatwebcando.today/

Complete List: https://developer.mozilla.org/en-US/docs/Web/API

See also: https://caniuse.com/

..

Speech API Demonstration: https://www.google.com/intl/en/chrome/demos/speech.html

Example: Local Storage (part of Web Storage API)

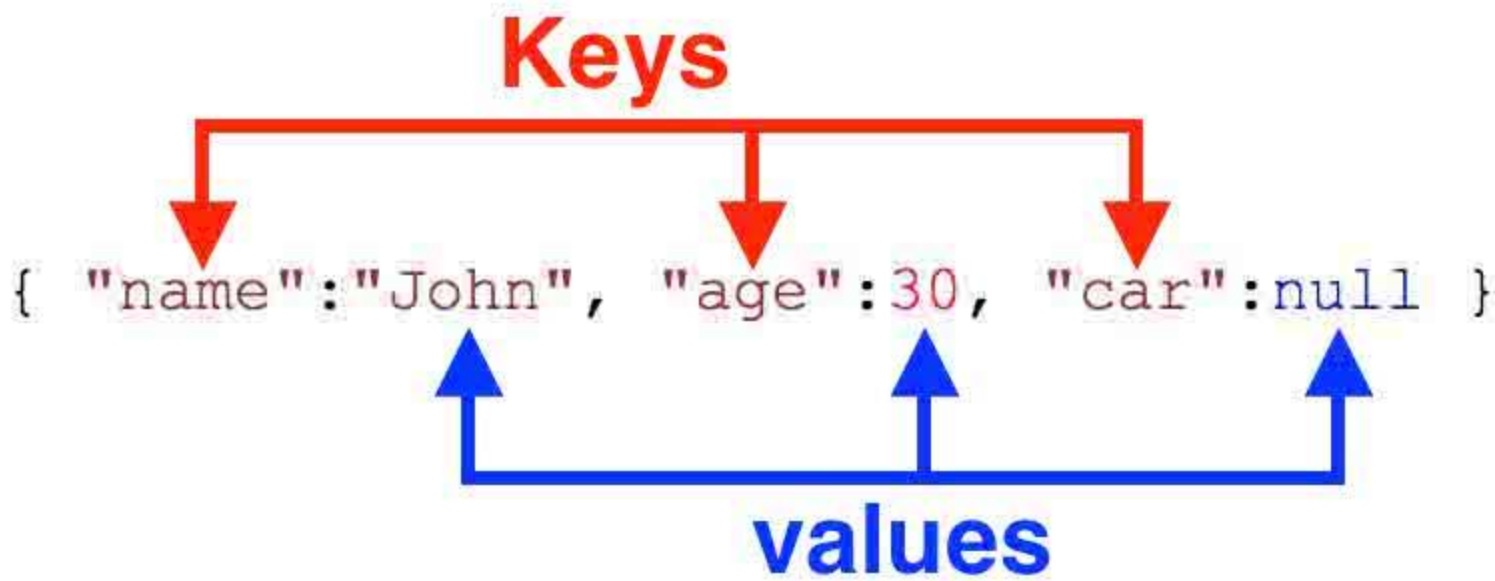
- Data is never transferred to the server.
- Stores data with no expiration date, and gets cleared only through JavaScript, or clearing the Browser cache.
- Storage limit 5..10 MB
- Data only accessible via Javacript

JSON (JavaScript Object Notation) is a set of text formatting rules for storing and transferring data in a machine and human readable way. It looks a lot like the object literal syntax of JavaScript, and it is from there JSON originates.

Douglas Crockford.

Specified JSON in 2000.

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "car": null,
  "age": 30,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
```


Keys
{ "name":"John", "age":30, "car":null }
values

Methods

- `JSON.stringify()` - converts a JavaScript object or value to a JSON string
- `JSON.parse()` - parses a JSON string, constructing the JavaScript value or object described by the string.

Let's build some servers now!

Further reading

- An Extensive Overview of Web Applications for Beginners. https://medium.com/@riteshs4hu/an-extensive-overview-of-web-applications-for-beginners-1d260dbc3675
- W3Schools. https://wwww.w3schools.org
- Learn to Code HTML and CSS. https://learn.shayhowe.com/html-css/
- HTML Guidelines. https://developer.mozilla.org/en-US/docs/MDN/Contribute/Guidelines/Code_guidelines/HTML
- Getting Started with HTML. https://htmldog.com/guides/html/beginner/gettingstarted
- The Modern Javascript tutorial: https://javascript.info/
- Learn Javascript online: https://learnjavascript.online/

Further reading

- HTML APIs in Depth https://medium.com/the-ui-girl/html-apis-in-depth-78f0abc918c8
- An overview of the device integration HTML5 APIs https://whatwebcando.today/
- MDN List of Web APIs: https://developer.mozilla.org/en-US/docs/Web/API
- Using HTML5 APIs in out web app https://levelup.gitconnected.com/using-html5-apis-in-our-web-app-f23b4aa974ac
- Short JSON Description: https://www.htmldog.com/guides/javascript/intermediate/json/
- How to Use JSON https://www.dummies.com/web-design-development/javascript/how-to-use-javascript-object-notation-json-for-html5-and-css3-progr

Thank you!